Contents

TCP Traffic

6 minute read ✓ page test

This task shows you how to set up Istio authorization policy for

TCP traffic in an Istio mesh.

Before you begin

Before you begin this task, do the following:

- Read the Istio authorization concepts.
- Install Istio using the Istio installation guide.
- Deploy two workloads named `sleep` and `tcp-echo` together in a namespace, for example `foo`. Both workloads run with an Envoy proxy in front of each. The `tcp-echo` workload listens on port 9000, 9001 and 9002 and echoes back any traffic it

received with a prefix `hello`. For example, if you send “world” to `tcp-echo`, it will reply with `hello world`. The `tcp-echo` Kubernetes service object only declares the ports 9000 and 9001, and omits the port 9002. A pass-through filter chain will handle port 9002 traffic. Deploy the example namespace and workloads using the following command:

```
$ kubectl create ns foo
$ kubectl apply -f <(istioctl kube-inject -f @samples/tcp-echo/tcp-echo.yaml@) -n foo
$ kubectl apply -f <(istioctl kube-inject -f @samples/sleep/sleep.yaml@) -n foo
```

- Verify that `sleep` successfully communicates with `tcp-echo`on ports 9000 and 9001 using the following command:

```
$ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o jsonpath={.it
ems..metadata.name})" -c sleep -n foo -- sh -c 'echo "port 9000" | nc
tcp-echo 9000' | grep "hello" && echo 'connection succeeded' || echo '
connection rejected'
hello port 9000
connection succeeded
```

```
$ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o jsonpath={.it
ems..metadata.name})" -c sleep -n foo -- sh -c 'echo "port 9001" | nc
tcp-echo 9001' | grep "hello" && echo 'connection succeeded' || echo '
connection rejected'
hello port 9001
connection succeeded
```

- Verify that `sleep` successfully communicates with `tcp-echo` on port 9002. You need to send the traffic directly to the pod IP of `tcp-echo` because the port 9002 is not defined in the Kubernetes service object of `tcp-echo`. Get the pod IP address and send the request with the following command:

```
$ TCP_ECHO_IP=$(kubectl get pod "$(kubectl get pod -l app=tcp-echo -n
foo -o jsonpath={.items..metadata.name})" -n foo -o jsonpath="{.status
.podIP}")
$ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o jsonpath={.it
ems..metadata.name})" -c sleep -n foo -- sh -c "echo \"port 9002\" | n
c $TCP_ECHO_IP 9002" | grep "hello" && echo 'connection succeeded' ||
echo 'connection rejected'
hello port 9002
connection succeeded
```

If you don’t see the expected output, retry after a few seconds. Caching and propagation can cause a delay.

Configure access control for a TCP workload

1. Create the `tcp-policy` authorization policy for the `tcp-echo` workload in the `foo` namespace. Run the following command to apply the policy to allow requests to port 9000 and 9001:

```
$ kubectl apply -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: tcp-policy
  namespace: foo
spec:
  selector:
    matchLabels:
      app: tcp-echo
  action: ALLOW
  rules:
  - to:
    - operation:
       ports: ["9000", "9001"]
EOF
```

2. Verify that requests to port 9000 are allowed using the following command:

```
$ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o jsonpath={.it
ems..metadata.name})" -c sleep -n foo -- sh -c 'echo "port 9000" | nc
tcp-echo 9000' | grep "hello" && echo 'connection succeeded' || echo '
connection rejected'
hello port 9000
connection succeeded
```

3. Verify that requests to port 9001 are allowed using the following command:

    ```
    $ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o jsonpath={.it
    ems..metadata.name})" -c sleep -n foo -- sh -c 'echo "port 9001" | nc
    tcp-echo 9001' | grep "hello" && echo 'connection succeeded' || echo '
    connection rejected'
    hello port 9001
    connection succeeded
    ```

4. Verify that requests to port 9002 are denied. This is enforced by the authorization policy which also applies to

the pass through filter chain, even if the port is not declared explicitly in the `tcp-echo` Kubernetes service object. Run the following command and verify the output:

```
$ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o jsonpath={.items..metadata.name})" -c sleep -n foo -- sh -c "echo \"port 9002\" | nc $TCP_ECHO_IP 9002" | grep "hello" && echo 'connection succeeded' || echo 'connection rejected'
connection rejected
```

5. Update the policy to add an HTTP-only field named `methods` for port 9000 using the following command:

```
$ kubectl apply -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: tcp-policy
  namespace: foo
spec:
  selector:
    matchLabels:
      app: tcp-echo
  action: ALLOW
  rules:
  - to:
    - operation:
        methods: ["GET"]
        ports: ["9000"]
EOF
```

6. Verify that requests to port 9000 are denied. This occurs because the rule becomes invalid when it uses an HTTP-

only field (`methods`) for TCP traffic. Istio ignores the invalid ALLOW rule. The final result is that the request is rejected, because it does not match any ALLOW rules. Run the following command and verify the output:

```
$ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o jsonpath={.it
ems..metadata.name})" -c sleep -n foo -- sh -c 'echo "port 9000" | nc
tcp-echo 9000' | grep "hello" && echo 'connection succeeded' || echo '
connection rejected'
connection rejected
```

7. Verify that requests to port 9001 are denied. This occurs because the requests do not match any ALLOW rules. Run the following command and verify the output:

```
$ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o jsonpath={.it
ems..metadata.name})" -c sleep -n foo -- sh -c 'echo "port 9001" | nc
tcp-echo 9001' | grep "hello" && echo 'connection succeeded' || echo '
connection rejected'
connection rejected
```

8. Update the policy to a DENY policy using the following command:

```
$ kubectl apply -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: tcp-policy
  namespace: foo
spec:
  selector:
    matchLabels:
      app: tcp-echo
  action: DENY
  rules:
  - to:
    - operation:
        methods: ["GET"]
        ports: ["9000"]
EOF
```

9. Verify that requests to port 9000 are denied. This occurs because Istio ignores the HTTP-only fields in an invalid

DENY rule. This is different from an invalid ALLOW rule, which causes Istio to ignore the entire rule. The final result is that only the `ports` field is used by Istio and the requests are denied because they match with the `ports`:

```
$ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o jsonpath={.it
ems..metadata.name})" -c sleep -n foo -- sh -c 'echo "port 9000" | nc
tcp-echo 9000' | grep "hello" && echo 'connection succeeded' || echo '
connection rejected'
connection rejected
```

10. Verify that requests to port 9001 are allowed. This occurs because the requests do not match the `ports` in the DENY policy:

```
$ kubectl exec "$(kubectl get pod -l app=sleep -n foo -o jsonpath={.it
ems..metadata.name})" -c sleep -n foo -- sh -c 'echo "port 9001" | nc
tcp-echo 9001' | grep "hello" && echo 'connection succeeded' || echo '
connection rejected'
hello port 9001
connection succeeded
```

Clean up

1. Remove the namespace foo:

```
$ kubectl delete namespace foo
```